IBM Data Science Capstone Project

Introduction

Hong Kong is a paradise for food lovers. It is famously known to be the culinary capital of Asia offering a wide variety of world's delicious food. In recent years, coffee culture has been brewing a storm and growing in popularity in Hong Kong. Hanging out in cafes became a popular trend among the younger generation. According to a market search, revenue in the coffee segment amounts to US$ 1,352 million in 2020 and the market is expected to grow annually by 8.1%.

With consumers' growing appreciation for coffee, more and more investors are motivated to open cafes in Hong Kong. In the brick-and-mortar retail world, it's said that the three most important decisions you'll make are location, location, and location. So putting the cafe in the proper location might be the single most important thing to do at startup. By using data science methods and machine learning techniques such as clustering, this project aims to identify the best location for running a cafe in Hong Kong.

Business Problem

The main idea behind the project is to help investors to analyse the optimal location for opening cafes in Hong Kong. However, opening a cafe can be challenging due to Hong Kong's high retail rents. Also, most business districts are now being awash with coffee shops. Starting a cafe business in such an area could be very competitive and won't be much profitable.Therefore, it is very important to find out the best possible neighborhood for opening a cafe.

Data acquisition

Following data sources will be needed to extract/generate the required information:

1. A list of the districts and neighborhoods in Hong Kong is obtained from the Rating and Valuation Department under the Government of Hong Kong: https://www.rvd.gov.hk/doc/tc/hkpr20/Appendix_TC.xlsx
2. Latitude and Longitude of these neighborhoods are retrieved via Geocoder API.
3. Top Venues data related to these neighborhoods is collected using Foursquare API

Data Cleaning

Hong Kong is divided into 18 Districts which are further divided into 127 sub-districts. The dataset available on the Government website is in Excel format (Appendix_TC.xlsx) but as we can see the data presentation was directly converted from the pdf format by the data owner:

	A	B	C	D	E
1					附錄 Appendix
2			各區域及地區 AREAS AND DISTRICTS		
3					
4					
5	區域 Area	地區 District	地區內的分區名稱	Names of Sub-districts within District Boundaries	小規劃統計區 Tertiary Planning Units
6					
7	港島 HONG KONG	中西區 Central and Western	堅尼地城、石塘咀、西營盤、上環、中環、金鐘、半山區、山頂	Kennedy Town, Shek Tong Tsui, Sai Ying Pun, Sheung Wan, Central, Admiralty, Mid-levels, Peak	111, 112, 113, 114, 115, 116, 121, 122, 123, 124(p), 141, 142, 143, 181, 182
8		灣仔 Wan Chai	灣仔、銅鑼灣、天后、跑馬地、大坑、掃桿埔、渣甸山	Wan Chai, Causeway Bay, Tin Hau, Happy Valley, Tai Hang, So Kon Po, Jardine's Lookout	124(p), 131, 132, 133, 134, 135, 140, 144, 145, 146, 147, 148(p), 149, 151(p), 152(p), 183, 184, 190
9		東區 Eastern	寶馬山、北角、鰂魚涌、西灣河、筲箕灣、柴灣、小西灣	Braemar Hill, North Point, Quarry Bay, Sai Wan Ho, Shau Kei Wan, Chai Wan, Siu Sai Wan	148(p), 151(p), 152(p), 153, 154, 155, 156, 157, 158, 161, 162, 163, 164, 165, 166, 167

After importing the dataset into Pandas dataframe, we found that the data is messy which contains **extra rows, unrelated columns, bogus \n** and **Chinese characters** that needs to be cleaned:

```
In [2]: # Import dataset into Pandas Dataframe

        hkn = pd.read_excel("https://www.rvd.gov.hk/doc/tc/hkpr20/Appendix_TC.xlsx", header = 4)
        hkn.head()
```

Out[2]:

	區域\nArea	地區\nDistrict	地區內的分區名稱	Names of Sub-districts\nwithin District Boundaries	小規劃統計區\nTertiary Planning Units
0	NaN	NaN	NaN	NaN	NaN
1	港島\nHONG KONG	中西區\nCentral and\nWestern	堅尼地城、石塘咀、\n西營盤、上環、\n中環、金鐘、\n半山區、山頂	Kennedy Town, Shek Tong Tsui,\nSai Ying Pun, Sheung Wan,\nCentral, Admiralty,\nMid-levels, Peak	111, 112, 113, 114, 115, 116, \n121, 122, 123, 124(p), 141,\n142, 143, 181, 182
2	NaN	灣仔\nWan Chai	灣仔、銅鑼灣、\n天后、跑馬地、大坑、\n掃桿埔、渣甸山	Wan Chai, Causeway Bay,\nTin Hau, Happy Valley, Tai Hang,\nSo Kon Po, Jardine's Lookout	124(p), 131, 132, 133, 134, 135, \n140, 144, 145, 146, 147, 148(p), \n149, 151(p), 152(p), 183, 184, \n190
3	NaN	東區\nEastern	寶馬山、北角、\n鰂魚涌、西灣河、\n筲箕灣、柴灣、\n小西灣	Braemar Hill, North Point,\nQuarry Bay, Sai Wan Ho,\nShau Kei Wan, Chai Wan,\nSiu Sai Wan	148(p), 151(p), 152(p), 153, \n154, 155, 156, 157, 158, 161, \n162, 163, 164, 165, 166, 167
4	NaN	南區\nSouthern	薄扶林、香港仔、\n鴨脷洲、黃竹坑、\n壽臣山、淺水灣、\n舂坎角、赤柱、\n大潭、石澳	Pok Fu Lam, Aberdeen,\nAp Lei Chau, Wong Chuk Hang, Shouson Hill, Repulse Bay,\nChung Hom Kok, Stanley,\nTai Tam, Shek O	171, 172, 173, 174, 175, 176, \n191, 192, 193, 194, 195, 196, \n197, 198

a) Remove unrelated columns

First of all, we use `del` keyword to completely remove unrelated columns such as "區域\nArea" , "地 區 內 的 分 區 名 稱" and "小 規 劃 統 計 區 \nTertiary Planning Units"

```
In [4]: # Remove unrelated columns

        del hkn ["區域\nArea"]
        del hkn["地區內的分區名稱"]
        del hkn["小規劃統計區\nTertiary Planning Units"]
        hkn.head()
```

Out[4]:

	地區\nDistrict	Names of Sub-districts\nwithin District Boundaries
0	NaN	NaN
1	中西區\nCentral and\nWestern	Kennedy Town, Shek Tong Tsui,\nSai Ying Pun, Sheung Wan,\nCentral, Admiralty,\nMid-levels, Peak
2	灣仔\nWan Chai	Wan Chai, Causeway Bay,\nTin Hau, Happy Valley, Tai Hang,\nSo Kon Po, Jardine's Lookout
3	東區\nEastern	Braemar Hill, North Point,\nQuarry Bay, Sai Wan Ho,\nShau Kei Wan, Chai Wan,\nSiu Sai Wan
4	南區\nSouthern	Pok Fu Lam, Aberdeen,\nAp Lei Chau, Wong Chuk Hang, Shouson Hill, Repulse Bay,\nChung Hom Kok, Stanley,\nTai Tam, Shek O

b) Rename columns

In [5]:
```python
# Simplify the column name

hkn.rename(columns={"地 區 \nDistrict" : "District",
                    "Names of Sub-districts\nwithin District Boundaries":"Neighborhood" } ,
          inplace = True)

hkn.head()
```

Out[5]:

	District	Neighborhood
0	NaN	NaN
1	中 西 區 \nCentral and\nWestern	Kennedy Town, Shek Tong Tsui,\nSai Ying Pun, Sheung Wan,\nCentral, Admiralty,\nMid-levels, Peak
2	灣 仔 \nWan Chai	Wan Chai, Causeway Bay,\nTin Hau, Happy Valley, Tai Hang,\nSo Kon Po, Jardine's Lookout
3	東 區 \nEastern	Braemar Hill, North Point,\nQuarry Bay, Sai Wan Ho,\nShau Kei Wan, Chai Wan,\nSiu Sai Wan
4	南 區 \nSouthern	Pok Fu Lam, Aberdeen,\nAp Lei Chau, Wong Chuk Hang, Shouson Hill, Repulse Bay,\nChung Hom Kok, Stanley,\nTai Tam, Shek O

c) Replace bogus \n with a spacing

After removing the bogus \n, the data looks more clean and tidy.

In [6]:
```python
#Replace bogus \n with spacing from data

hkn = hkn.replace('\n',' ', regex=True)
hkn.head()
hkn.tail(15)
```

Out[6]:

	District	Neighborhood
24	NaN	NaN
25	NaN	NaN
26	NaN	NaN
27	NaN	NaN
28	NaN	小 規 劃 統 計 區 Tertiary Planning Units
29	NaN	NaN
30	NaN	113, 114, 115
31	NaN	121, 122, 123, 124
32	NaN	131, 132, 133, 134, 135, 144, 145, 146, 147, 149
33	NaN	151, 152, 153, 154, 155, 156, 157
34	NaN	211, 212, 213, 214, 215, 216, 217
35	NaN	220, 221, 222, 225, 226, 227, 228, 229, 251, 252, 253, 256
36	NaN	NaN
37	NaN	NaN
38	NaN	NaN

d) Remove all empty rows

Now take a look at the bottom 15 rows of the dataframe, some rows are completely empty. To solve this problem, we use Pandas `notnull()` method to find out the rows are not empty and the result is then stored in the dataframe.

In [15]:
```
# Method 1 - to find out not null values in the Neighborhood column
df2 = pd.notnull(hkn["Neighborhood"])
# only take those not null value
hkn = hkn[df2]

# we want to remove the rows if the data is nan in District column
df3 = pd.notnull(hkn["District"])
hkn = hkn[df3]


# Method 2 - For loop: to remove the empty rows (do not contain any data)
# check all rows if it is null (True = null, False = not null), store the result in df2

# removed_elements = []
# df2 = pd.isnull(hkn)

# remove the rows if both District and Neighborhood are null

# for n in range(len(hkn)) :
#     if df2.at[n,'District'] and df2.at[n, 'Neighborhood'] :
#         removed_elements.append(n)

# hkn.drop(removed_elements, axis = 0, inplace = True)

# hkn
```

In [16]:
```
hkn
```

Out[16]:

	District	Neighborhood
1	中西區 Central and Western	Kennedy Town, Shek Tong Tsui, Sai Ying Pun, Sheung Wan, Central, Admiralty, Mid-levels, Peak
2	灣仔 Wan Chai	Wan Chai, Causeway Bay, Tin Hau, Happy Valley, Tai Hang, So Kon Po, Jardine's Lookout
3	東區 Eastern	Braemar Hill, North Point, Quarry Bay, Sai Wan Ho, Shau Kei Wan, Chai Wan, Siu Sai Wan
4	南區 Southern	Pok Fu Lam, Aberdeen, Ap Lei Chau, Wong Chuk Hang, Shouson Hill, Repulse Bay, Chung Hom Kok, Stanley, Tai Tam, Shek O
7	油尖旺 Yau Tsim Mong	Tsim Sha Tsui, Yau Ma Tei, West Kowloon Cultural District, King's Park, Mong Kok, Tai Kok Tsui
8	深水埗 Sham Shui Po	Mei Foo, Lai Chi Kok, Cheung Sha Wan, Sham Shui Po, Shek Kip Mei, Yau Yat Tsuen, Tai Wo Ping, Stonecutters Island
9	九龍城 Kowloon City	Hung Hom, To Kwa Wan, Ma Tau Kok, Ma Tau Wai, Kai Tak, Kowloon City, Ho Man Tin, Kowloon Tong, Beacon Hill
10	黃大仙 Wong Tai Sin	San Po Kong, Wong Tai Sin, Tung Tau, Wang Tau Hom, Lok Fu, Diamond Hill, Tsz Wan Shan, Ngau Chi Wan
11	觀塘 Kwun Tong	Ping Shek, Kowloon Bay, Ngau Tau Kok, Jordan Valley, Kwun Tong, Sau Mau Ping, Lam Tin, Yau Tong
14	葵青 Kwai Tsing	Kwai Chung, Tsing Yi
15	荃灣 Tsuen Wan	Tsuen Wan, Sheung Kwai Chung, Ting Kau, Sham Tseng, Tsing Lung Tau, Ma Wan, Sunny Bay
16	屯門 Tuen Mun	Tai Lam Chung, So Kwun Wat, Tuen Mun, Lam Tei
17	元朗 Yuen Long	Hung Shui Kiu, Ha Tsuen, Lau Fau Shan, Tin Shui Wai, Yuen Long, San Tin, Lok Ma Chau, Kam Tin, Shek Kong, Pat Heung
18	北區 North	Fanling, Luen Wo Hui, Sheung Shui, Shek Wu Hui, Sha Tau Kok, Luk Keng, Wu Kau Tang
19	大埔 Tai Po	Tai Po Market, Tai Po, Tai Po Kau, Tai Mei Tuk, Shuen Wan, Cheung Muk Tau, Kei Ling Ha
20	沙田 Sha Tin	Tai Wai, Sha Tin, Fo Tan, Ma Liu Shui, Wu Kai Sha, Ma On Shan
21	西貢 Sai Kung	Clear Water Bay, Sai Kung, Tai Mong Tsai, Tseung Kwan O, Hang Hau, Tiu Keng Leng, Ma Yau Tong
22	離島 Islands	Cheung Chau, Peng Chau, Lantau Island, (including Tung Chung, Discovery Bay), Lamma Island

e) Remove Chinese characters

We import string library string.printable function to filter out all sets of punctuation, digits, ascii_letters and whitespace.

```
In [10]: # Remove Chinese characters in the dataframe

         import string

         printable = set(string.printable)
         hkn['District'] = hkn['District'].apply(lambda row: ''.join(filter(lambda x: x in printable, row)))
```

f) Split each of the Neighborhoods into a new row

In the Neighborhood column, multiple neighborhoods are placed in the same row. We have to split it into a new row under the same district. We use three methods to handle this case. Firstly, we use assign() method to assign a new column called "Neighborhood" to hkn. Then, we use str.split() method to split strings on a given separator (" , ") in the hkn["Neighborhood"] column. Lastly, we use explode() method to transform each element of a list-like to a row, replicating index values. Therefore you may already be aware that the index values are the same within the same district.

```
In [19]: # to split the Neighborhood
         hkn = hkn.assign(Neighborhood=hkn['Neighborhood'].str.split(',')).explode('Neighborhood')


         # The other method to split the Neighborhood

         # hkn = (hkn.set_index(hkn.columns.drop('Neighborhood',1).tolist())
         #   .Neighborhood.str.split(',', expand=True)
         #   .stack()
         #   .reset_index()
         #   .rename(columns={0:'Neighborhood'})
         #   .loc[:, hkn.columns]
         # )
```

```
In [21]: hkn.head
```

```
Out[21]: <bound method NDFrame.head of                  District                    Neighborhood
         1       Central and Western  Kennedy Town
         1       Central and Western   Shek Tong Tsui
         1       Central and Western   Sai Ying Pun
         1       Central and Western   Sheung Wan
         1       Central and Western   Central
         1       Central and Western   Admiralty
         1       Central and Western   Mid-levels
         1       Central and Western   Peak
         2      Wan Chai              Wan Chai
         2      Wan Chai               Causeway Bay
         2      Wan Chai               Tin Hau
         2      Wan Chai               Happy Valley
         2      Wan Chai               Tai Hang
         2      Wan Chai               So Kon Po
         2      Wan Chai               Jardine's Lookout
         3      Eastern               Braemar Hill
         3      Eastern                North Point
         3      Eastern                Quarry Bay
         3      Eastern                Sai Wan Ho
```

g) Reset the index of the dataframe

As we have duplicated index values for the same district, we have to reset the index for further analysis. We use `reset_index()` method and we use drop parameter `drop = True` to avoid the old index being added as a column.

```
In [13]: # reset the index
         hkn = hkn.reset_index(drop = True)
         hkn
```

Out[13]:

	District	Neighborhood
0	Central and Western	Kennedy Town
1	Central and Western	Shek Tong Tsui
2	Central and Western	Sai Ying Pun
3	Central and Western	Sheung Wan
4	Central and Western	Central
5	Central and Western	Admiralty
6	Central and Western	Mid-levels
7	Central and Western	Peak
8	Wan Chai	Wan Chai
9	Wan Chai	Causeway Bay
10	Wan Chai	Tin Hau

```
In [14]: # Check if the dataframe contains any missing values
         hkn.isnull().values.any()

Out[14]: False
```

Finally, we check if there are any missing values in the dataframe.

Methodology

First of all, we download the local districts and neighborhoods dataset from the Hong Kong government department web page, then clean and extract the data into a Pandas DataFrame.

Add Geo-coordinates to each district

Before we want to get the top venues data from Foursquare, we get the geographical coordinates (i.e. Latitude and Longitude) of the neighborhoods using the Geocoder package (https://geocoder.readthedocs.io/index.html) which converts addresses into geographic coordinates. After collecting the data, we will populate the data into a Pandas DataFrame.

In [15]:
```
# Get the coordinates of each Neighborhood
hkn["Coordinates"] = hkn["Neighborhood"].apply(geolocator.geocode)

# Use apply lambda function to get the latitude and longitude and store it in respective columns
hkn["Latitude"] = hkn["Coordinates"].apply(lambda x: x.latitude if x!= None else None)
hkn["Longitude"] = hkn["Coordinates"].apply(lambda x: x.longitude if x!= None else None)
hkn
```

Out[15]:

	District	Neighborhood	Coordinates	Latitude	Longitude
0	Central and Western	Kennedy Town	(堅尼地城 Kennedy Town, 士美菲路 Smithfield, 摩星嶺 Mount Davis, 堅尼地城 Kennedy Town, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.28131165, 114.12916039816602))	22.281312	114.129160
1	Central and Western	Shek Tong Tsui	(石塘咀 Shek Tong Tsui, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.287735, 114.1345987))	22.287735	114.134599
2	Central and Western	Sai Ying Pun	(西營盤 Sai Ying Pun, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.286121, 114.1420862))	22.286121	114.142086
3	Central and Western	Sheung Wan	(上環 Sheung Wan, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.2868701, 114.150267))	22.286870	114.150267
4	Central and Western	Central	(Central, Venezuela, (9.577626, -68.42482001504955))	9.577626	-68.424820

We got two problems here. The first one is an empty row due to the typo in the neighborhood and therefore cannot get coordinates. The other is that some of the neighborhoods are obviously mistakenly located via Geocoder. We fix it by concatenating the neighborhood with “Hong Kong, China” and store it in a newly created column called “Address” so as to specify the country for measuring the geo-coordinates precisely.

In [20]:
```
# In case the district is not found via Geocoder, we can concatenate the neighborhood with the area to form a detailed
hkn["Address"] = hkn["Neighborhood"] + "," + "Hong Kong, China"
```

In [22]:
```
# Get the coordinates of each district
hkn["Coordinates"] = hkn["Address"].apply(geolocator.geocode)

# Use apply lambda function to get the latitude and longitude and store it in respective columns
hkn["Latitude"] = hkn["Coordinates"].apply(lambda x: x.latitude if x!= None else None)
hkn["Longitude"] = hkn["Coordinates"].apply(lambda x: x.longitude if x!= None else None)
hkn
```

Out[22]:

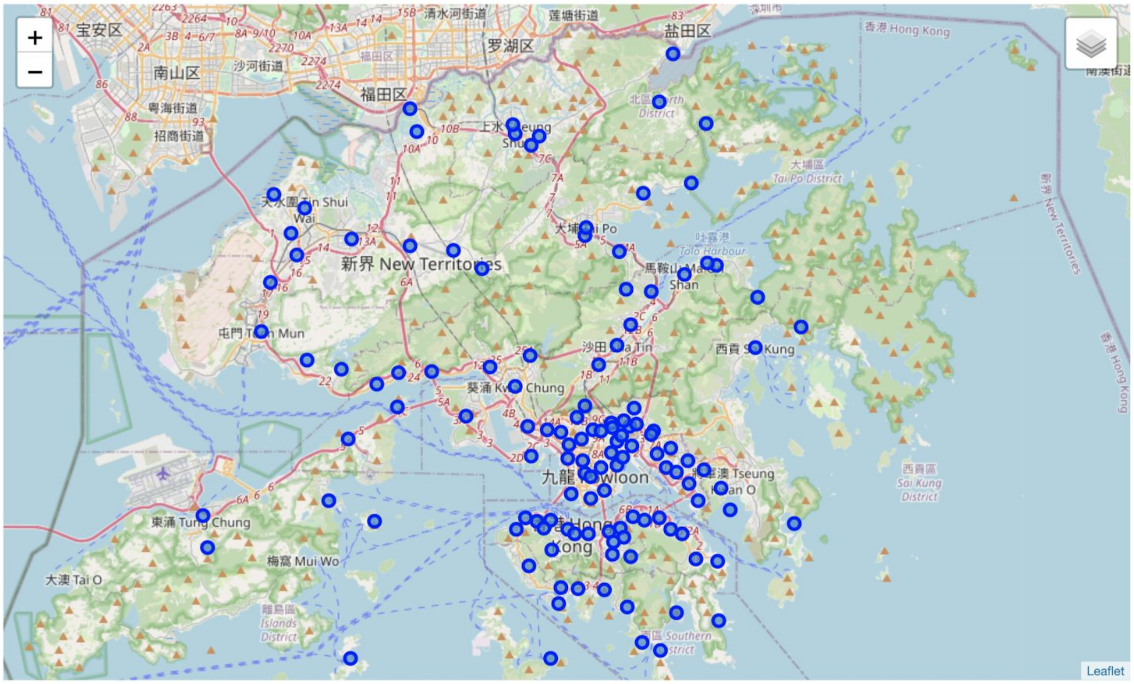
	District	Neighborhood	Coordinates	Latitude	Longitude	Address
0	Central and Western	Kennedy Town	(堅尼地城 Kennedy Town, 士美菲路 Smithfield, 摩星嶺 Mount Davis, 堅尼地城 Kennedy Town, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.28131165, 114.12916039816602))	22.281312	114.129160	Kennedy Town,Hong Kong, China
1	Central and Western	Shek Tong Tsui	(石塘咀 Shek Tong Tsui, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.287735, 114.1345987))	22.287735	114.134599	Shek Tong Tsui,Hong Kong, China
2	Central and Western	Sai Ying Pun	(西營盤 Sai Ying Pun, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.286121, 114.1420862))	22.286121	114.142086	Sai Ying Pun,Hong Kong, China
3	Central and Western	Sheung Wan	(上環 Sheung Wan, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.2868701, 114.150267))	22.286870	114.150267	Sheung Wan,Hong Kong, China
4	Central and Western	Central	(中環 Central, 西環 Sai Wan, 香港島 Hong Kong Island, 中西區 Central and Western District, 香港 Hong Kong, China 中国, (22.2813223, 114.1602579))	22.281322	114.160258	Central,Hong Kong, China

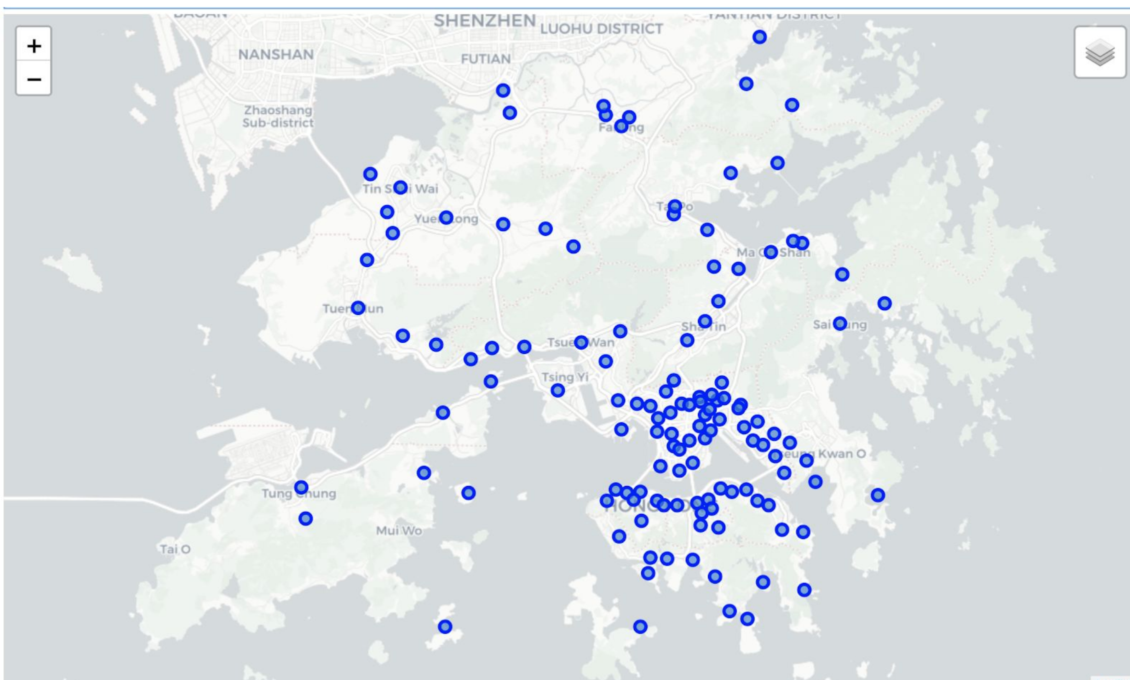
As we get all the coordinates of each neighborhood correctly, we may remove "Coordinates" and "Address" columns now.

Visualize neighborhoods on map

Then, we visualize the center locations of each neighborhood in a map using the Folium package. The map of Hong Kong is created with districts superimposed on top.This allows us to perform a sanity check to make sure that the geographical coordinates data returned by Geocoder are correctly plotted in Hong Kong. We add a layer control button for displaying different tiles (cartodbpositron and openstreetmap) in the same map.

Out[26]:


宝安区
南山区
罗湖区
福田区
盐田区
新界 New Territories
東涌 Tung Chung
梅窩 Mui Wo
大澳 Tai O
Leaflet


SHENZHEN
LUOHU DISTRICT
NANSHAN
FUTIAN
Zhaoshang
Sub-district
Tsuen Wan
Tsing Yi
Mui Wo
Tai O
Leaflet

Exploring Top Venues for each neighborhood with Foursquare API

In order to explore the surroundings in the neighborhoods, we use Foursquare explore venue API to access and acquire the venue data such as venue name, venue unique ID, venue category, venue location (latitude and longitude) etc. for those neighborhoods. We need to register a Foursquare Developer account so as to obtain credentials (ie. client ID and client Secret key).

Define Foursquare Credentials and Version

```
In [27]: CLIENT_ID = 'I0AVIIA5NFB1R5PB1FYFVRGM1NZ42CLK15IVDFFYKUJX1WTZ' # your Foursquare ID
         CLIENT_SECRET = '1G4MRZXUS04SOYWMFTCZJP5KC5PGEDBSMWLFORMZD3D4UTUM' # your Foursquare Secret
         VERSION = '20200522' # Date of Today

         print('Your credentails:')
         print('CLIENT_ID: ' + CLIENT_ID)
         print('CLIENT_SECRET:' + CLIENT_SECRET)

         Your credentails:
         CLIENT_ID: I0AVIIA5NFB1R5PB1FYFVRGM1NZ42CLK15IVDFFYKUJX1WTZ
         CLIENT_SECRET:1G4MRZXUS04SOYWMFTCZJP5KC5PGEDBSMWLFORMZD3D4UTUM
```

Afterward, we make API calls (request) to Foursquare passing in the geographical coordinates of the neighborhoods in a Python loop. Take Hong Kong as an example. To simplify the results, we set the LIMIT property as 100 and radius as 1000.

```
In [30]: radius = 1000
         LIMIT = 100

         # Define the corresponding URL to get the venues data from Foursquare API
         url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.fc
             CLIENT_ID,
             CLIENT_SECRET,
             VERSION,
             neighborhood_latitude,
             neighborhood_longitude,
             radius,
             LIMIT)
         url

Out[30]: 'https://api.foursquare.com/v2/venues/explore?&client_id=I0AVIIA5NFB1R5PB1FYFVRGM1NZ42CLK15IVDFFYKUJX1WTZ&client_secr
         et=1G4MRZXUS04SOYWMFTCZJP5KC5PGEDBSMWLFORMZD3D4UTUM&v=20200522&ll=22.28131165,114.12916039816602&radius=1000&limit=10
         0'
```

Foursquare will return the venue data up to 100 venues for a coordinate in JSON format. We need to transform JSON files into a Pandas DataFrame and filter out the data except the venue's name, categories, latitude and longitude.

```
In [31]: # Send the GET Request to Foursquare and the results will be returned in JSON format

         results = requests.get(url).json()
         results
```


```
Out[31]: {'meta': {'code': 200, 'requestId': '5ee0ad39f89b1820a6198998'},
          'response': {'suggestedFilters': {'header': 'Tap to show:',
            'filters': [{'name': 'Open now', 'key': 'openNow'}]},
           'headerLocation': 'Sai Wan',
           'headerFullLocation': 'Sai Wan, Hong Kong',
           'headerLocationGranularity': 'neighborhood',
           'totalResults': 123,
           'suggestedBounds': {'ne': {'lat': 22.290311659000007,
             'lng': 114.13886847158932},
            'sw': {'lat': 22.27231164099999, 'lng': 114.11945232474272}},
           'groups': [{'type': 'Recommended Places',
             'name': 'recommended',
             'items': [{'reasons': {'count': 0,
                'items': [{'summary': 'This spot is popular',
                  'type': 'general',
                  'reasonName': 'globalInteractionReason'}]},
               'venue': {'id': '5ac99cc067af3a34ce55398b',
                'name': 'Winstons Coffee',
                'location': {'address': 'Shop 8, G/F, The Hudson, 11 Davis St',
```


We analyse each neighborhood by creating a function to repeat the same process to all the neighborhoods. As a result, a total of 3,049 venues in the neighborhoods are identified.

```
In [37]: print(hk_venues.shape)
         hk_venues.head()

         (3049, 7)
```

Out[37]:

	Neighborhood	N_Latitude	N_Longitude	Venue	V_Latitude	V_Longitude	Venue Category
0	Kennedy Town	22.281312	114.12916	Winstons Coffee	22.281374	114.127172	Coffee Shop
1	Kennedy Town	22.281312	114.12916	Sun Hing Restaurant (新興食家)	22.283036	114.128209	Dim Sum Restaurant
2	Kennedy Town	22.281312	114.12916	Comptoir	22.281209	114.126975	French Restaurant
3	Kennedy Town	22.281312	114.12916	Little Creatures	22.283950	114.128264	Brewery
4	Kennedy Town	22.281312	114.12916	Kyo Japanese (京日本料理)	22.281499	114.127110	Japanese Restaurant

We group the rows by Neighborhood to count the total number of venues in each neighborhood . We also found out that there are 267 unique categories.

In [39]:
```
hk_venues.groupby('Neighborhood').count()
```

Out[39]:

Neighborhood	N_Latitude	N_Longitude	Venue	V_Latitude	V_Longitude	Venue Category
Aberdeen	22	22	22	22	22	22
Admiralty	58	58	58	58	58	58
Ap Lei Chau	17	17	17	17	17	17
Beacon Hill	2	2	2	2	2	2
Causeway Bay	71	71	71	71	71	71
Central	97	97	97	97	97	97
Chai Wan	26	26	26	26	26	26
Cheung Muk Tau	2	2	2	2	2	2
Cheung Sha Wan	21	21	21	21	21	21
Chung Hom Kok	2	2	2	2	2	2

Checking how many distinct venue categories we have

In [40]:
```
print('There are {} uniques categories.'.format(len(hk_venues['Venue Category'].unique())))
```

```
There are 267 uniques categories.
```

Analyzing the Districts

Before performing K-means clustering algorithms on the data, we need to create one-hot encoding to the venue category and take the mean of each category for every neighborhood.

In [43]:
```
hk_grouped = hk_onehot.groupby('Neighborhood').mean().reset_index()
hk_grouped
```

Out[43]:

	Neighborhood	Zoo	ATM	Accessories Store	Airport Service	American Restaurant	Arcade	Arepa Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Astrolog
0	Aberdeen	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.045455	0.0000
1	Admiralty	0.017241	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000
2	Ap Lei Chau	0.000000	0.000000	0.000000	0.000000	0.058824	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000
3	Beacon Hill	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000
4	Causeway Bay	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000
5	Central	0.010309	0.000000	0.000000	0.010309	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.010309	0.0000
6	Chai Wan	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000
7	Cheung Muk Tau	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000
8	Cheung Sha ...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.0000

In [44]:
```
hk_grouped.shape
```

Out[44]:
```
(124, 267)
```

In total, there are 124 rows as well as neighborhoods grouped in the dataframe. The size of the grouped dataframe is different from the neighborhood dataframe (n=127). We found that the missing neighborhoods are “Stonecutters Island”, “Tai Lam Chung” and “Pat Heung”. The result shows that there are three places missing in the grouped dataframe.

```
In [45]: missing_neighborhood = [i for i in hkn['Neighborhood'].unique() if i not in hk_grouped['Neighborhood'].unique()]
         missing_neighborhood
Out[45]: [' Stonecutters Island', 'Tai Lam Chung', ' Pat Heung']
```

As far as we know, Stonecutters Island is a military port, while Tai Lam Chung is a country park where it is famous for the Tai Lam Chung Reservoir. Lastly, Pat Heung is a rural area without business activities. Therefore, it is a good idea to exclude these places from the dataset.

Finally, we get the top 5 most common venues together with their frequency for each neighborhood.

```
In [48]: # Print each neighborhood along with the top 5 most common venues.

         num_top_venues = 5

         for hood in hk_grouped['Neighborhood']:
             print("----"+hood+"----")
             temp = hk_grouped[hk_grouped['Neighborhood'] == hood].T.reset_index()
             temp.columns = ['venue','freq']
             temp = temp.iloc[1:]
             temp['freq'] = temp['freq'].astype(float)
             temp = temp.round({'freq': 2})
             print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
             print('\n')

         ---- Aberdeen----
                           venue  freq
         0  Athletics & Sports   0.09
         1  Sushi Restaurant     0.09
         2  Supermarket          0.09
         3  Noodle House         0.05
         4  Pharmacy             0.05


         ---- Admiralty----
                              venue  freq
         0  Café                     0.10
         1  Hotel                    0.09
         2  Park                     0.05
         3  Italian Restaurant       0.05
         4  Vietnamese Restaurant    0.03


         ---- Ap Lei Chau----
```

Now we create a new dataframe to display the top 10 most common venues for each neighborhood. As the data pre-processing is completed, we start running k-means clustering.

In [50]:
```python
# Create the new dataframe and display the top 10 venues for each neighborhood.

num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = hk_grouped['Neighborhood']

for ind in np.arange(hk_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(hk_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

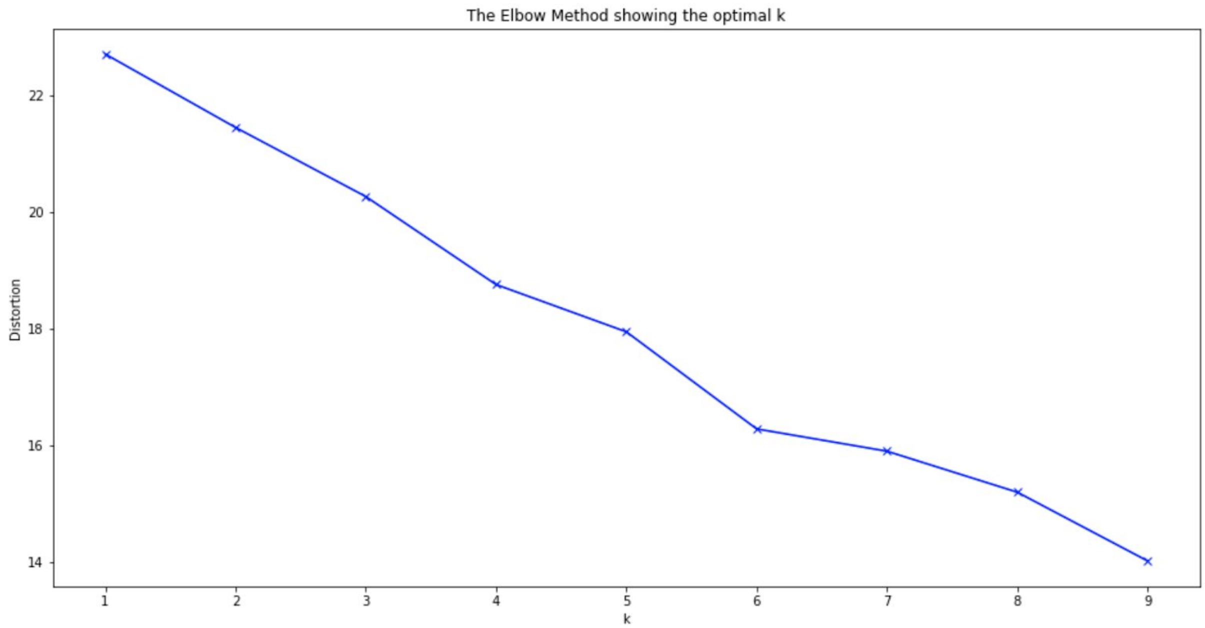
Out[50]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Aberdeen	Supermarket	Sushi Restaurant	Athletics & Sports	Cha Chaan Teng	Asian Restaurant	Electronics Store	Fast Food Restaurant	Shopping Mall	Bus Station	Chinese Restaurant
1	Admiralty	Café	Hotel	Park	Italian Restaurant	Tea Room	Seafood Restaurant	Vietnamese Restaurant	Steakhouse	Yoga Studio	Gourmet Shop
2	Ap Lei Chau	Fast Food Restaurant	Chinese Restaurant	Shopping Mall	Pet Store	Restaurant	Paper / Office Supplies Store	Café	American Restaurant	Hotel	Mountain
3	Beacon Hill	Scenic Lookout	Mountain	Fish Market	English Restaurant	Farm	Farmers Market	Fast Food Restaurant	Field	Fish & Chips Shop	Flea Market
4	Causeway Bay	Japanese Restaurant	Sushi Restaurant	Chinese Restaurant	Hotel	Dessert Shop	Gift Shop	Shopping Mall	Bakery	Szechuan Restaurant	Clothing Store

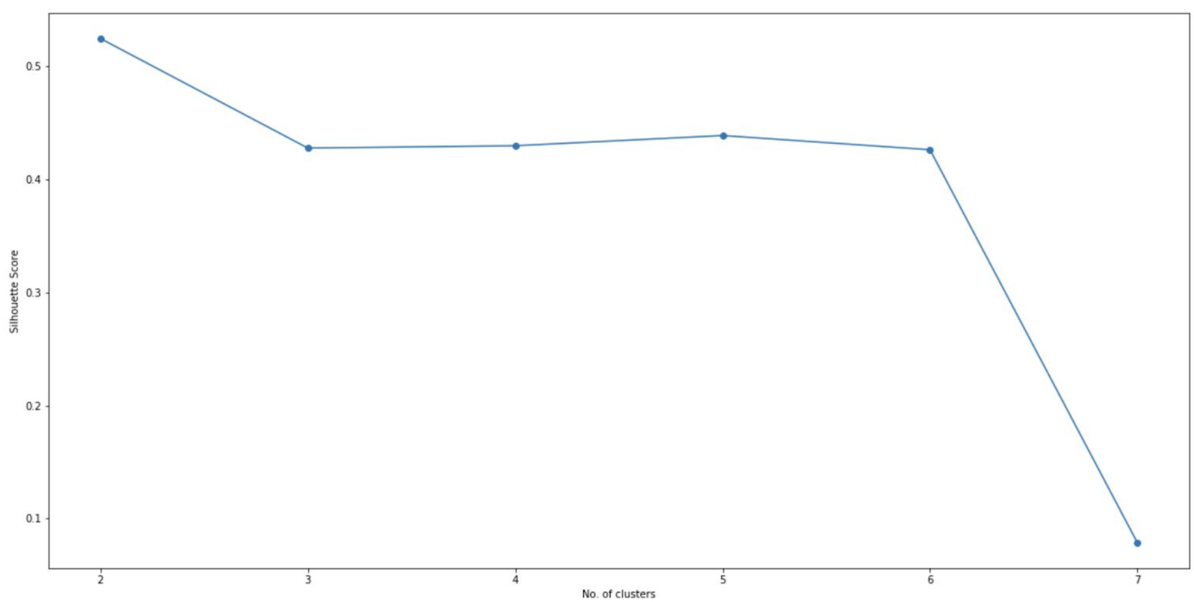
Using Machine Learning for Clustering Neighborhoods

In this project, our goal is to identify the best location for running a cafe. As we already know the most common venues in each neighborhood, we use K-means clustering algorithm for venue segmentation. It is an unsupervised algorithm to partition n observations into k clusters that have similar characteristics.

In the first step, we find the optimal number of k for running k-means clustering by using the Elbow Method. The Elbow Method is a very popular technique and the idea is to run k-means clustering for a range of clusters k (let's say from 1 to 10) and for each value, we are calculating the sum of squared distances from each point to its assigned center (distortions). When the distortions are plotted and the plot looks like an arm then the "elbow"(the point of inflection on the curve) is the best value of k.



We can observe that the Elbow Method does not have enough evidence to show the optimal k. Instead, we use Silhouette Score to find out that the optimal number of clusters is 6.



We run the K-means clustering algorithm to cluster the neighborhood into 6 clusters, suggested by the result above.

Run k-means to cluster the neighborhood into 6 clusters.

In [53]:
```
# set number of clusters
kclusters = 6

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(hk_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[53]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 0], dtype=int32)
```

Results

The results will allow us to identify which neighborhoods have higher concentration of restaurants while which have fewer.

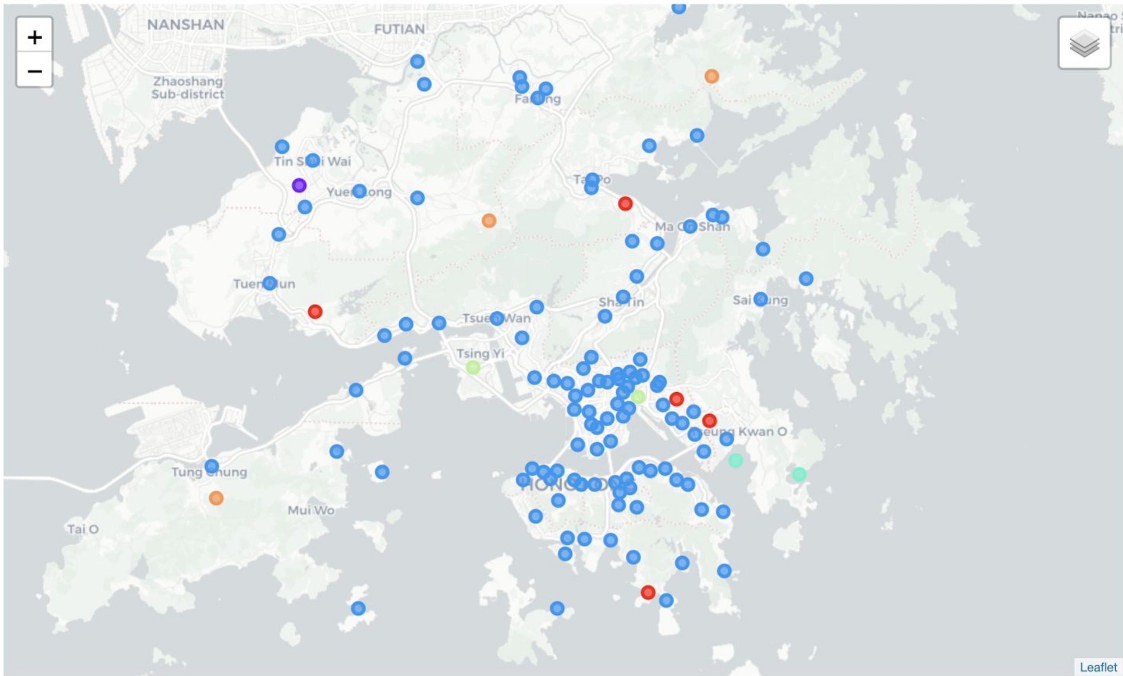
In [54]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

In [55]:
```
hk_merged = hkn

hk_merged = hk_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

hk_merged
```

Out[55]:

	District	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue
0	Central and Western	Kennedy Town	22.281312	114.129160	2.0	Japanese Restaurant	Coffee Shop	Hong Kong Restaurant	Mexican Restaurant	Vietnamese Restaurant	Chinese Restaurant	Park
1	Central and Western	Shek Tong Tsui	22.287735	114.134599	2.0	Noodle House	Chinese Restaurant	Malay Restaurant	Pier	Supermarket	Boxing Gym	French Restaurant
2	Central and Western	Sai Ying Pun	22.286121	114.142086	2.0	Coffee Shop	Hotel	French Restaurant	Chinese Restaurant	Noodle House	Supermarket	Burger Joint
3	Central and Western	Sheung Wan	22.286870	114.150267	2.0	Japanese Restaurant	Café	Coffee Shop	French Restaurant	Chinese Restaurant	Italian Restaurant	Bar
4	Central and Western	Central	22.281322	114.160258	2.0	Chinese Restaurant	Steakhouse	Social Club	Sushi Restaurant	Gym / Fitness Center	Lounge	French Restaurant

Out[61]:



Cluster 1

In [62]:
```
hk_merged.loc[hk_merged['Cluster Labels'] == 0, hk_merged.columns[[1] + list(range(5, hk_merged.shape[1]))]]
```

Out[62]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
28	Chung Hom Kok	Park	Beach	Zhejiang Restaurant	Fish & Chips Shop	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market	Fujian Restaurant
66	Jordan Valley	Fast Food Restaurant	Park	Zhejiang Restaurant	Electronics Store	Fried Chicken Joint	French Restaurant	Food Court	Food & Drink Shop	Food	Flea Market
81	So Kwun Wat	Cha Chaan Teng	Zhejiang Restaurant	Fish & Chips Shop	English Restaurant	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market	Eastern European Restaurant
103	Tai Po Kau	Park	BBQ Joint	Restaurant	Zhejiang Restaurant	Fish & Chips Shop	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market
120	Ma Yau Tong	Convenience Store	Cha Chaan Teng	Park	Zhejiang Restaurant	Fish & Chips Shop	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market

Cluster 2

In [63]:
```
hk_merged.loc[hk_merged['Cluster Labels'] == 1, hk_merged.columns[[1] + list(range(5, hk_merged.shape[1]))]]
```

Out[63]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
85	Ha Tsuen	Hong Kong Restaurant	Zhejiang Restaurant	Fish & Chips Shop	English Restaurant	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market	Eastern European Restaurant

Cluster 3

```
In [64]: hk_merged.loc[hk_merged['Cluster Labels'] == 2, hk_merged.columns[[1] + list(range(5, hk_merged.shape[1]))]]
```

Out[64]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Kennedy Town	Japanese Restaurant	Coffee Shop	Hong Kong Restaurant	Mexican Restaurant	Vietnamese Restaurant	Chinese Restaurant	Park	Fish & Chips Shop	French Restaurant	Italian Restaurant
1	Shek Tong Tsui	Noodle House	Chinese Restaurant	Malay Restaurant	Pier	Supermarket	Boxing Gym	French Restaurant	Burger Joint	Furniture / Home Store	Spanish Restaurant
2	Sai Ying Pun	Coffee Shop	Hotel	French Restaurant	Chinese Restaurant	Noodle House	Supermarket	Burger Joint	Tapas Restaurant	Park	Hotpot Restaurant
3	Sheung Wan	Japanese Restaurant	Café	Coffee Shop	French Restaurant	Chinese Restaurant	Italian Restaurant	Bar	Thai Restaurant	Grocery Store	Tapas Restaurant
4	Central	Chinese Restaurant	Steakhouse	Social Club	Sushi Restaurant	Gym / Fitness Center	Lounge	French Restaurant	Coffee Shop	Hotel	Gym
5	Admiralty	Café	Hotel	Park	Italian Restaurant	Tea Room	Seafood Restaurant	Vietnamese Restaurant	Steakhouse	Yoga Studio	Gourmet Shop
6	Mid-levels	Thai	Japanese	Tapas	Noodle	Café	Coffee Shop	Seafood	Korean	Tea Room	Beer Store

Cluster 4

```
In [65]: hk_merged.loc[hk_merged['Cluster Labels'] == 3, hk_merged.columns[[1] + list(range(5, hk_merged.shape[1]))]]
```

Out[65]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
114	Clear Water Bay	Harbor / Marina	Eastern European Restaurant	Fried Chicken Joint	French Restaurant	Food Court	Food & Drink Shop	Food	Flea Market	Fish Market	Fish & Chips Shop
117	Tseung Kwan O	Harbor / Marina	Eastern European Restaurant	Fried Chicken Joint	French Restaurant	Food Court	Food & Drink Shop	Food	Flea Market	Fish Market	Fish & Chips Shop

Cluster 5

```
In [66]: hk_merged.loc[hk_merged['Cluster Labels'] == 4, hk_merged.columns[[1] + list(range(5, hk_merged.shape[1]))]]
```

Out[66]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
50	Kai Tak	Tunnel	Metro Station	Zhejiang Restaurant	Fish & Chips Shop	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market	Electronics Store
72	Tsing Yi	Tunnel	Zhejiang Restaurant	Fish & Chips Shop	English Restaurant	Farm	Farmers Market	Fast Food Restaurant	Field	Fish Market	Eastern European Restaurant

Cluster 6

```
In [69]: hk_merged.loc[hk_merged['Cluster Labels'] == 5, hk_merged.columns[[1] + list(range(5, hk_merged.shape[1]))]]
```

Out[69]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
92	Shek Kong	Trail	Farm	Fish & Chips Shop	Electronics Store	English Restaurant	Farmers Market	Fast Food Restaurant	Field	Zhejiang Restaurant	Fujian Restaurant
100	Wu Kau Tang	Trail	Other Great Outdoors	Waterfall	Zhejiang Restaurant	Field	Electronics Store	English Restaurant	Farm	Farmers Market	Fast Food Restaurant
123	Lantau Island	Trail	Fish & Chips Shop	Electronics Store	English Restaurant	Farm	Farmers Market	Fast Food Restaurant	Field	Zhejiang Restaurant	Fujian Restaurant

Discussion and Limitations

In this project, we focus on the most common venues in the neighborhoods as well as the frequency of occurrence of coffee shops. In order to make an insightful data-driven business decision, there are other factors for consideration such as retail rent rate, population and income of residences, the concentration of commercial buildings as well as offices that could influence the location decision for running a cafe in Hong Kong. Such data are published in different channels which make it difficult to collect and integrate.

Future research could devise a methodology to estimate such data to be used in the clustering algorithm to identify the prime locations for opening cafes. Last but not least, this project uses a free Sandbox Tier account of Foursquare API that came with limitations on how many API calls and results returned. Future research could use a paid account to bypass these limitations for better results.

Conclusion

By looking at the cluster data, we can see that cluster 3 is the one that we are the most interested in. In this cluster, the majority of the most common venues are food and restaurant. We can conclude that the best location is indicated in cluster 3. However a big cluster it is, we can perform an in-death analysis taking into account the rent rate, population and income of residences and other factors to find out the most potential neighborhood for running cafes business. The rest of the clusters (cluster 1: Park and fastfood Restaurant, cluster 2: Hong Kong Restaurant, cluster 4: Harbor / Marina, cluster 5: Tunnel and cluster 6: Trail) reflected their local specialties in the districts which were considered not to be an ideal place for running cafes business.